ORACLE

MAAソリューションの継続的サービスのためのアプリケーション・チェックリスト

バージョン[1.0]

公開

ORACLE

目次

ORACLE

はじめに

以下のチェックリストは、アプリケーションの継続的な可用性を実現するようにお使いの環境を準備する際に役立ちます。アプリケーション・コンティニュイティがお使いのデータベース・サービスで有効になっていない場合、あるいはお使いのアプリケーションで使用しない場合でも、このホワイト・ペーパーに記載されている内容には、可用性を継続できるようにシステムを準備する上で大きな価値があります。

手順は段階的に実施でき、次のような要素で構成されています。

- データベース・サービスの使用
- URLまたは接続文字列における高可用性の構成
- 高速アプリケーション通知（FAN）の有効化
- ドレイニングに対応する推奨プラクティスの使用
- アプリケーション・コンティニュイティまたは透過的アプリケーション・コンティニュイティの有効化

このチェックリストの主な対象読者は、アプリケーション開発者およびアプリケーション所有者です。また、DBAおよびPDB管理者向けに運用例を記載しています。

ORACLE

データベース・サービスの使用

サービスとは、作業を管理するためのロジックを抽象化したものです。サービスは、単一のシステム・イメージによって作業を管理するため、基盤システムの複雑さはクライアントからは分かりません。アプリケーションは、高可用性機能（FAN、ドレイニング、アプリケーション・コンティニュイティ）を使用してサービスに接続しなければなりません。このサービスは、デフォルトのデータベース・サービスやデフォルトのPDBサービス（データベースやPDBと同じ名前のサービス）ではありません。

サービスに関するサーバー側の手順

サービスを作成するには、以下に示すようなコマンドを使用します。

複数のサイトを使用している場合、プライマリ・サイトではプライマリ・ロールを使用して、Oracle Active Data Guardで管理されるセカンダリ・サイトで開かれるサービスではスタンバイ・ロールを使用してサービスを作成する必要があります。サービスは、ロールに基づきサイトで自動的に起動および停止されます。

基本的なサービスの作成

```
$ srvctl add service -db mydb -service MYSERVICE –preferred inst1 -available inst2 -pdb mypdb -notification
TRUE -drain_timeout 300 -stopoption IMMEDIATE
-role PRIMARY
```

透過的アプリケーション・コンティニュイティ

```
$ srvctl add service -db mydb -service TACSERVICE -pdb mypdb –preferred inst1
-available inst2 -failover_restore AUTO -commit_outcome TRUE -failovertype AUTO -
replay_init_time 600 -retention 86400 -notification TRUE -drain_timeout
300 -stopoption IMMEDIATE -role PRIMARY
```

アプリケーション・コンティニュイティ

```
$ srvctl add service -db mydb -service ACSERVICE -pdb mypdb -preferred inst1 - available inst2 -
failover_restore LEVEL1 -commit_outcome TRUE -failovertype TRANSACTION -session_state dynamic -
replay_init_time 600 -retention 86400 - notification TRUE -drain_timeout 300 -stopoption IMMEDIATE -role
PRIMARY
```

TAF Select Plus

```
$ srvctl add service -db mydb -service TAFSERVICE -pdb mypdb -preferred inst1 - available inst2 -
failover_restore LEVEL1 -commit_outcome TRUE -failovertype SELECT - - notification TRUE -drain_timeout
300 -stopoption TRANSACTIONAL -role PRIMARY
```

注：

接続文字列でRETRY_COUNTおよびRETRY_DELAYが推奨どおり設定されている場合は、failoverretryおよびfailoverdelayは必要ありません。また、ここでも表示していません。

19c以降では、利用可能な状態の優先インスタンスを使用するサービスについて、srvctlでフェイルバック・オプション（-failback）が使用できるようになりました。優先インスタンスが利用可能になると、そのインスタンスで実行されていないサービスはフェイルバックします。データベース・セッションで不要な停止が発生するため、高可用性目的では、このオプションは推奨されません。サービスは、ロケーションについて透過的である必要があります。ただし、優先インスタンスでのサービスを必要とするデプロイメントでは、フェイルバック・オプションが役に立つ場合があります。

ORACLE

URLまたは接続文字列における高可用性の構成

オラクルでは、基本的な起動、フェイルオーバー、スイッチオーバー、およびフォールバックのときに正常に接続されるようにするため、アプリケーションで次の接続文字列の構成を使用することを推奨しています。

RETRY_COUNT、RETRY_DELAY、CONNECT_TIMEOUT、およびTRANSPORT_CONNECT_TIMEOUTパラメータを設定し、接続リクエスト時にサービスが利用可能になるまで待って、正常に接続されるようにします。これらの値を、お使いのMAAソリューションに応じて、あらゆるOracle Real Application Clusters（Oracle RAC）ノード障害とOracle Data Guardロール移行でフェイルオーバーが許可されるようにチューニングします。

ルール：（詳細については、「アプリケーションとサーバーのタイムアウトの整合」を参照） RETRY_COUNTを使用する場合は、常にRETRY_DELAYを設定します。
(RETRY_COUNT +1) * RETRY_DELAYがOracle RACおよびData Guardのリカバリ時間の最大値よりも大きくなるように設定します。

低速なWide Area Networkを使用している場合を除いて、TRANSPORT_CONNECT_TIMEOUTを1～5秒の範囲に設定します。

CONNECT_TIMEOUTを高い値に設定して、ログイン・ストームを回避します。値を低くすると、アプリケーションまたはプールのキャンセルや接続の再試行のために、ログイン'合戦'が発生することがあります。

EZCONNECTではFANの自動構成機能が回避されるため、クライアントで簡易接続ネーミングを使用しないでください。

待機時間は、センチ秒（cs）またはミリ秒（ms）のいずれかの単位で設定できます。デフォルトの単位は秒（s）です。

LDAPやtnsnames.oraといった中央のロケーションで接続文字列やURLを保守します。プロパティ・ファイルやプライベート・ロケーションに接続文字列やURLを分散させないでください。保守が極めて困難になります。一元管理されるロケーションを使用すると、標準の形式、チューニング、サービス設定を保持するのに役立ちます。

以下は、12.2以降のすべてのOracleドライバに推奨される接続文字列です。特定の値がチューニングされる場合もありますが、最初はこの例で示す値を使用すると良いでしょう。

```
Alias (or URL) = (DESCRIPTION = (CONNECT_TIMEOUT=
90)(RETRY_COUNT=50)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
  (ADDRESS_LIST =
     (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521))) (ADDRESS_LIST =
     (LOAD_BALANCE=on)
     (ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

高速アプリケーション通知（FAN）の有効化

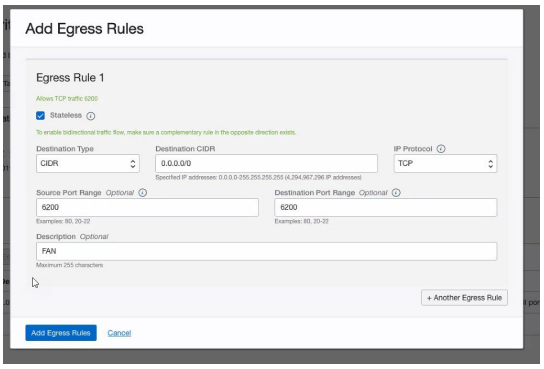
FANは、アプリケーションのフェイルオーバーを中断する場合の必須コンポーネントです。ノードまたはネットワークが停止した場合は、アプリケーションをリアルタイムで中断する必要があります。FANを有効化しておかないと、ノード、ネットワーク、サイトの障害など、ハードの物理的障害が発生した場合にアプリケーションがハングします。

Oracle Database 19cおよびOracleクライアント19c以降、FANでは次の2つの重要な機能強化が図られています。

- 計画イベントでは、FANは帯域内でドライバに直接送信されます。Javaの場合、バグ31112088の修正が必要になります。
- Oracle DatabaseおよびOracleクライアント・ドライバは、接続テストおよびリクエスト境界でFANの受信時にドレインします。

サーバー側

RAC、RAC-One、ADGのノードでFANポートを開く必要があります。これは、ADB-D、EXA、ExaCCを対象とするために非常に重要です。以下に例を示します。



クライアント側

FANを使用するためのコード変更はありません。Oracle Database 12c、Oracleクライアント12c、およびOracle Grid Infrastructure 12c以降、FANは標準で自動構成され、有効化されます。Oracleデータベースに接続されると、OracleデータベースはURLまたはTNS接続文字列を読み取り、クライアントでFANを自動構成します。上記のTNS形式を使用することは、FANを自動構成する上で重要です（別の形式を使用すると、FANを自動構成できなくなります）。FANを使用するには、データベース・サービスに接続し（手順1）、Oracle Notification Service（ONS）からイベントを受信できるようにする必要があります。

FANイベントの受信を監視するには、以下のホワイト・ペーパーで説明しているFANWatcherユーティリティを使用できます。

http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/fastapplic ationnotification12c-2538999.pdf

上述の自動構成の方法でFANWatcherユーティリティがONSにサブスクライブして、受信時にイベントを出力できる場合は、アプリケーションでもそれができるということになります。FANWatcherがこのタスクを実行できない場合は、次の操作を実行してください。

- （Grid Infrastructureクラスタの）ONSポートがファイアウォールによってブロックされていないことを確認します
- アプリケーションからのクライアント・サブスクリプションが作成されていることを確認します　詳細については、「*FAN*のデバッグ」セクションを参照してください。

クライアント固有の追加手順については、クライアントの構成を参照してください。なお、FANを使用するためにアプリケーション・コードを変更する必要はありません。

ドレイニングに対応する推奨プラクティスの使用

ベスト・プラクティスに従って計画メンテナンスを行う場合は、アプリケーション サーバーを再起動する必要はありません。

計画メンテナンスを行う場合は、メンテナンスを開始する前に、進行中の作業を完了させる時間を確保することを推奨します。作業をドレイニングすることでこれを行います。ドレイニングには次のように複数の方法があります。アプリケーションにもっとも適した方法を選択してください。

- Oracle接続プール
- 標準ドライバ側の接続テスト
- サーバー側の接続テストおよびさらに多くの接続ルール
- 透過的アプリケーション・コンティニュイティによる、予定のフェイルオーバー

ドレイニング用に割り当てられた時間内に完了しないリクエストに対しては、選択したフェイルオーバー・ソリューションと組み合わせてドレイニングを使用します。割り当てられた時間内にドレイニングが予定されていないセッションについては、フェイルオーバー・ソリューションによってリカバリが試行されます。

Oracle接続プールの使用

計画メンテナンス中だと認識させないための推奨されるソリューションとしては、Oracle接続プールを使用する方法を推奨します。アプリケーションでOracleプールが使用され、リクエストとリクエストの間にプールに接続が戻される場合、メンテナンス時にユーザーに影響はありません。Oracle Universal Connection Pool（Oracle UCP）、WebLogic Active GridLink、Oracle Tuxedo、OCIセッション・プール、ODP.NET管理対象プロバイダおよび管理対象外プロバイダなどのOracleプールがサポートされます。ドレイニングするためにアプリケーション変更などを行う必要は一切ありません。使用と使用の間に必ず接続をプールに返却することのみが必要です。接続テストの有効化も推奨されます。

サード・パーティ・アプリケーション・サーバーとOracle UCP、またはリクエスト境界のない接続プールとOracle UCPの使用

サード・パーティのJavaベースのアプリケーション・サーバーを使用している場合、もっとも効果的にドレイニングやフェイルオーバーを行うには、プールされたデータソースの代わりにOracle UCPを使用します。このアプローチは、Oracle WebLogic Server、IBM WebSphereおよびIBM Liberty（XA使用およびXA不使用）、Apache Tomcat、Hikari CPなど、多くのアプリケーション・サーバーでサポートされています。Oracle UCPをデータソースとして使用すると、高速接続フェイルオーバー、ランタイム・ロードバランシング、アプリケーション・コンティニュイティといった、十分に動作保証されたOracle UCPの機能を使用できます。

Red Hat JBossでアプリケーションがJ2EEまたはContainer Managed Transactions（CMT）を使用している場合、Red Hat JBoss EAP 7.4にはリクエスト境界が設定されます。この構成では、FAN（XA使用およびXA不使用）を使用したドレイニング、およびアプリケーション・コンティニュイティ（XA不使用）を使用したドレイニングをサポートします。

注：接続プールへの接続の返却

アプリケーションは、各リクエストの最後に接続を接続プールに返却する必要があります。アプリケーションで必要になったときだけ接続をチェックアウトするのがベスト・プラクティスです。接続をプールに返却せずに保持し続けても、パフォーマンス面のメリットはありません。したがってアプリケーションでは、接続をチェックアウトし、その後処理が完了したらすぐにその接続をチェックインします。そうすることで、他のスレッドや、自分のスレッドで再び必要になったときに接続を使用できます。良好なパフォーマンスを得るため、接続を接続プールに返却することが一般的に推奨されます。

アプリケーションをドレイニングする接続テストの使用

Oracleプールを使用できない場合、Oracleクライアント・ドライバ19cまたはOracle Database 19cでは、セッションがドレイニングされます。サービスが再配置された場合や停止した場合、またはOracle Data Guard経由でスタンバイ・サイトにスイッチオーバーされた場合、Oracle DatabaseとOracleクライアント・ドライバには、以下に基づいて、接続をリリースするための安全な場所を検索するよう通知されます。

- 接続性を検証する標準接続テスト
- ドレイニングのためにサーバーで事前定義されたルール

- 接続性を検証するカスタムSQLテスト

接続テストの失敗を理由に、さらに多くのプロセスを停止しないでください。接続テストは、その接続に適用されます。接続プールは、接続を閉じて別の接続を取得します。

Thin JDBCドライバによる標準接続テストの使用

JDBCドライバによる接続テストを使用するには、以下を実行します。

- ValidateConnectionOnBorrow = trueに設定します
- Javaシステム・プロパティを設定します
  - -Doracle.jdbc.fanEnabled=true（Oracle ADB-Sのみの場合はfalseを使用）
  - -Doracle.jdbc.defaultConnectionValidation=SOCKET（オプション）

それから、以下のテストを設定します。java.sql.Connection.isValid(int timeout)

注：失敗セッション・オプションはプールにのみ使用し、接続テスト障害時はプールのフラッシュと破棄を無効化してください。

OCIドライバによる接続テストの使用

OCIドライバを直接使用する場合は、OCI_ATTR_SERVER_STATUSを使用します。これは、コードを変更する唯一の方法です。コードにおいて、接続の借用時と返却時にサーバー・ハンドルを確認し、セッションが切断されているかどうかを確かめます。サービスが停止または再配置されると、OCI_ATTR_SERVER_STATUSの値がOCI_SERVER_NOT_CONNECTEDに設定されます。OCIセッション・プールを使用している場合、この接続チェックが行われます。

以下のコード・サンプルは、OCI_ATTR_SERVER_STATUSの使用方法を示しています。

```
ub4 serverStatus = 0
OCIAttrGet((dvoid *)srvhp, OCI_HTYPE_SERVER,
                        (dvoid *)&serverStatus, (ub4 *)0, OCI_ATTR_SERVER_STATUS, errhp);
if (serverStatus == OCI_SERVER_NORMAL)
        printf("Connection is up.\n");
else if (serverStatus == OCI_SERVER_NOT_CONNECTED)
          printf("Connection is down.\n");
```

Oracle Databaseへの接続テストの使用

Oracleプールを使用できない場合、Oracle Database 19cではセッションをドレイニングできます。

有効化した接続テストや接続ルールは、ビューDBA_CONNECTION_TESTSで確認できます。SQLベースの接続テストを使用する場合は、接続プールまたはアプリケーション・サーバーにおいて、データベースで有効化されているSQLと同じもの（同じ文）を使用します。

追加の接続テストが必要な場合は、サービス、プラガブル・データベース、非コンテナ・データベースへの接続テストを追加、削除、有効化、または無効化できます。たとえば、次のとおりです。

```
SQL> EXECUTE
        dbms_app_cont_admin.add_sql_connection_test('SELECT COUNT(1) FROM DUAL');
SQL> EXECUTE n
SQL> SET LINESIZE 120
SQL> SELECT * FROM DBA_CONNECTION_TESTS
```

注：接続テストでは、DBRU19.10以降のリリース・アップデートとともにリリースされた、バグ31863118の修正（すべてのSQLドライニングに適用可能）が必要です。

USERENVを使用したPLSQLワークロードのドレイニング

アプリケーションが、長時間実行されているPLSQLを使用している場合、userenv関数を使用して、セッションがドレイニング・モードであるかどうかを判定します。たとえば、長時間実行されているPL/SQLループ内でバッチ間のPLSQLブロックを終了するためのチェックにこの関数を使用します。この機能は、Oracle Database 19cのリリース・アップデート10より利用できます。

```
SQL> select SYS_CONTEXT('USERENV', 'DRAIN_STATUS') from dual ;

YS_CONTEXT('USERENV','DRAIN_STATUS')
--------------------------------------------------------------------------------
DRAINING

SQL> select SYS_CONTEXT('USERENV', 'DRAIN_STATUS') from dual ;

SYS_CONTEXT('USERENV','DRAIN_STATUS')
--------------------------------------------------------------------------------
NONE
```

透過的アプリケーション・コンティニュイティ（TAC）による、予定のフェイルオーバーの使用

Oracle Database 19cでは、アプリケーション・コンティニュイティに対し、予定のフェイルオーバーを導入しました。TACによって検出されるアプリケーションでは、フェッチおよびクリアでカーソルをクローズして、Oracleの複雑なPLSQL状態を使用しません。これらのアプリケーションでは、計画停止時および計画外停止時のフェイルオーバーにおいて、TACによる予定のフェイルオーバーが標準で使えるソリューションです。

メンテナンス実行中、新規リクエストが開始され、TACによって黙示的な境界が検出されると、予定のフェイルオーバーが実行されます。予定のフェイルオーバーは、SQL*PLUSによって使用されます。予定のフェイルオーバーは、概してSELECTS、INSERTS、UPDATES、DELETESを使用するアプリケーションでメリットがあります。（ヒント：Oracle 19cの場合のみ、SERVEROUTPUTは設定しないでください。）

この機能は、TACまたはACを使用している場合、Oracle Database 19cのOCIクライアント、およびJDBC Thinクライアント19RU12で有効化されています。

TAF SELECT Plusの使用

旧バージョンのOCIベース構成ではプリコンパイラ（PRO*C、PRO*COBOL）またはOracle ODBCを使用しているものや、OCIのアプリケーション・コンティニュイティでまだサポートされていないOCI APIを使用しているものがあります。このような旧バージョンのOCIベース・アプリケーションでの予定のフェイルオーバーには、TAF SELECT PLUSがドレイニングに適したオプションとなる場合があります。TAF SELECT PLUSを使用するには、別途サービスを作成して、サービス属性FAILOVER_TYPE=SELECT, FAILOVER_RESTORE=LEVEL1, COMMIT_OUTCOME=TRUE.

ドレイニングのためのサーバー側の手順

Oracle Databaseに接続するサービスは、接続テストを行うように構成し、ドレイニングに使用できる時間をdrain_timeoutに指定し、ドレイニング・タイムアウトの終了後に適用されるstopoptionにIMMEDIATEを指定します。SRVCTLで管理する停止、再配置、スイッチオーバーの各コマンドには、drain_timeoutおよびstopoptionスイッチが含まれており、サービスで設定された値を必要に応じてオーバーライドできます。

メンテナンス用のコマンドは、以下で説明するコマンドと類似していますOracle Fleet Patching and Provisioning（Oracle FPP）などのOracleツールでは、これらのコマンドを使用します。これらのコマンドを使用してドレイニングを開始します。必要に応じて追加オプションを含めます。詳しくは、My Oracle Support（MOS）Note：Doc ID 1593712.1を参照してください。

たとえば、RACのメンテナンスを実行するインスタンスを停止するには、以下に示すコマンドを使用します。再配置可能なサービスは再配置されます。CRSは、現在実行されていないインスタンスを開始する場合があり、そのインスタンスを要求するサービスを実行することができます。再配置できないサービス、または再配置を必要としないサービスは停止されます。

1. 他に"利用可能な"インスタンスなしでシングルトン・サービスが定義されている場合、予期された動作として完全な停止時間が生じる可能性があります。優先インスタンスを常に使用することを推奨します。

   ヒント：これらをスクリプトで使用している場合、*wait = yes*の追加が役に立つ場合があります。

2. RACインスタンスの再起動後、Clusterwareサービス属性がサービスの終了場所を自動的に決定するため、追加のsrvctlアクションは必要ありません。

たとえば、ドレイニングを行うインスタンスを停止するには、以下を実行します。

```
srvctl stop instance -db <db_name> -node <node_name> -stopoption immediate – drain_timeout <#> -force -failover

srvctl stop instance -db <db_name> -node <node_name> -stopoption immediate – drain_timeout <#> -force -failover -role primary
```

データベース、ノードまたはPDB別にすべてのサービスを再配置するには、以下を実行します。

```
srvctl relocate service -database <db_unique_name> -oldinst <old_inst_name> [- newinst <new_inst_name>] -drain_timeout <timeout> -stopoption <stop_option> - force

srvctl relocate service -database <db_unique_name> -currentnode <current_node> [- targetnode <target_node>] -drain_timeout <timeout> -stopoption <stop_option> - force

srvctl relocate service -database <db_unique_name> -pdb <pluggable_database> {- oldinst <old_inst_name> [-newinst <new_inst_name>] | -currentnode <current_node> [-targetnode <target_node>]} -drain_timeout <timeout> -stopoption <stop_option>
-force
```

*inst1*という名前のインスタンス（任意のインスタンス）で*GOLD*という名前のサービスを停止するには、以下を実行します。

```
srvctl stop service –db myDB –service GOLD –instance inst1 -drain_timeout <timeout> -stopoption <stop_option>
```

Data Guard Brokerを使用し、待ち時間60秒のタイムアウトでData Guardのセカンダリ・サイトにスイッチオーバーするには、以下を実行します。

```
SWITCHOVER TO dg_south WAIT 60
```

Data Guard Brokerを使用し、サービスから取得した待ち時間のタイムアウトでData Guardのセカンダリ・サイトにスイッチオーバーするには、以下を実行します。

```
SWITCHOVER TO dg_south WAIT
```

アプリケーション・コンティニュイティまたは透過的アプリケーション・コンティニュイティの有効化

アプリケーション・コンティニュイティは、アプリケーションがドレイニングされない場合のフェイルオーバー、タイムアウトの処理、および計画外停止の処理において強く推奨されます。アプリケーション・コンティニュイティは、次に示す2つの構成のいずれかを使用してデータベース・サービスで有効化されます。

アプリケーション・コンティニュイティ（AC）

アプリケーション・コンティニュイティ（AC）は、停止を認識させないようにする機能です。Oracleデータベース12.1以降、JavaベースのThinアプリケーションで使用でき、Oracle Database 12.2では、OCIおよびODP.NETベースのアプリケーションでも使用できます。Oracle Database 19c以降は、オープンソース・ドライバ（Node.js、Pythonなど）がサポートされます。アプリケーション・コンティニュイティを有効にすると、セッションの状態やトランザクションの状態などが分かっている時点からセッションがリカバリされて再構築されます。処理中のすべての作業がアプリケーション・コンティニュイティによって再構築されます。フェイルオーバーが行われると実行時間がわずかに長くなりますが、アプリケーションはそれまでと同様に動作し続けます。アプリケーション・コンティニュイティの標準モードは、Oracle接続プールを使用したOLTPアプリケーションに適しています。

透過的アプリケーション・コンティニュイティ（TAC）

Oracle Database 19c以降、セッションとトランザクションの状態を透過的に追跡および記録する透過的アプリケーション・コンティニュイティ（TAC）機能が導入され、リカバリ可能な停止の後にデータベース・セッションをリカバリできるようになりました。このリカバリにはアプリケーションの知識もアプリケーションのコード変更も必要ありません。そのため、現行のアプリケーションのTACを有効にできます。アプリケーションを透過的にフェイルオーバーするため、アプリケーションからユーザー・コールが発行されたときのセッション状態の使用状況を取得して分類した状態追跡情報が使用されます。

アプリケーション・コンティニュイティ（AC/TAC）を使用する場合の手順

接続プールへの接続の返却

リクエスト境界は、アプリケーション・コンティニュイティでは必須であり、透過的アプリケーション・コンティニュイティでは推奨されています。Oracle Universal Connection Pool（Oracle UCP）やOCIセッション・プールといったOracle接続プール、ODP.Net管理対象外プロバイダ、またはWebLogic Active GridLinkやRedHat JBossを使用している場合は、リクエスト境界が埋め込まれています。アプリケーションで使用する接続は、リクエストのたびにOracle接続プールに返却してリクエスト境界を取得する必要があります。

さらに、透過的アプリケーション・コンティニュイティでは、リクエスト境界が検出されます。Oracle Database 19cで境界が検出される条件は以下のとおりです。

- 進行中のトランザクションが存在しない。
- カーソルがフェッチ状態に残っていない。
- リストア不可能なセッション状態が存在しない（PLSQLグローバル、OJVM、移入された一時表）。
  サービス上でRESET_STATEを使用することで、これらは消去されます。

サービスでのFAILOVER_RESTORE

データベース・サービスで属性FAILOVER_RESTOREを設定する必要があります。ACではFAILOVER_RESTORE=LEVEL1を、TACではFAILOVER_RESTORE=AUTOを使用します。変更可能なすべてのパラメータは、ウォレットとFAILOVER_RESTOREを使用して自動的にリストアされるようになりました（『Oracle Real Application Clusters』の「アプリケーション・コンティニュイティの確保」を参照してください）。

ウォレットはADBDおよびADBS用に有効化されており、これらのウォレットはデータベース・リンクで使用されているものと同じです。接続確立時とフェイルオーバー時にカスタムの値をさらに構成するには、以下を使用します。

- ログオン・トリガー
- 接続初期化コールバックもしくはOracle UCPラベル（Javaの場合）、またはTAFコールバック（OCI、ODP.Netの場合）
- Oracle UCPまたはOracle WebLogic Serverの接続ラベル付け

関数の元の値のリストア

Oracle関数の元の結果を維持する機能は、SYSDATE、SYSTIMESTAMP、SYS_GUID、sequence.NEXTVAL、CURRENT_TIMESTAMP、およびLOCALTIMESTAMP向けに提供されています。SQLの所有シーケンスで識別シーケンスがサポートされます。元の値が保持されず、再実行時に異なる値がアプリケーションに返される場合は、再実行が拒否されます。

Oracle Database 19cでは、SQLの元の値が自動的に維持されます。PLSQLを使用している場合、アプリケーション・ユーザーにはGRANT KEEPを使用し、シーケンス所有者にはKEEP句を使用します。

たとえば、次のとおりです。

```
SQL> GRANT KEEP DATE TIME to scott;

SQL> GRANT KEEP SYSGUID to scott;

SQL> GRANT KEEP SEQUENCE mySequence on mysequence.myobject to scott;
```

副次的影響

データベース・リクエストに電子メール送信やファイル送信などの外部コールが含まれている場合、これは副次的影響です。再実行時に副次的影響を再実行するかどうかを選択します。多くのアプリケーションでは、ジャーナルの入力や電子メールの送信といった副次的影響を再実行する必要があります。アプリケーション・コンティニュイティでは、副次的影響は再実行されます（disableReplayを使用して無効化することができます）。一方、透過的アプリケーション・コンティニュイティはデフォルトで有効化されているため、TACでは副次的影響は再実行されません。

監視

アプリケーション・コンティニュイティでは、統計情報が収集されて、保護レベルが監視されます。これらの統計情報は自動ワークロード・リポジトリに保存され、自動ワークロード・リポジトリ・レポートで確認できます。

次の統計情報を、問合せで使用できます。

```
Statistic
----------------------------------------------------------

cumulative begin requests
cumulative end requests
cumulative user calls in requests
cumulative user calls protected by Application Continuity
successful replays by Application Continuity
rejected replays by Application Continuity
cumulative DB time protected in requests
```

たとえばサービス別の保護履歴のレポートを作成するには、以下を実行します。

```
set pagesize 60
set lines 120
col Service_name format a30 trunc heading "Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"
col time_protected format 999.999 heading "Time Prot"

select con_id, service_name, total_requests,
total_calls,total_protected,time_protected,total_protected*100/NULLIF(total_cal
ls,0) as Protected
from(
select * from
(select a.con_id, a.service_name, c.name,b.value
    FROM    gv$session a, gv$sesstat b, gv$statname c
    WHERE   a.sid           = b.sid
    AND     a.inst_id       = b.inst_id
    AND     b.value         != 0
    AND     b.statistic#    = c.statistic#
    AND     b.inst_id       = c.inst_id
    AND     a.service_name not in ('SYS$USERS','SYS$BACKGROUND'))
pivot(
    sum(value)
    for name in ('cumulative begin requests' as total_requests, 'cumulative end
requests' as Total_end_requests, 'cumulative user calls in requests' as
Total_calls, 'cumulative DB time protected in requests' as time_protected,
'cumulative user calls protected by Application Continuity' as total_protected)
))
order by con_id, service_name;
```

次のような形式の出力が表示されます。

CON_ID	Service	Requests	Calls in requests	Calls Protected	Time Prot	Protected %
109	RDDAINSUH6U1OKC_TESTY_high.adb	11	7			63
	RDDAINSUH6U1OKC_TESTY_tp.adb.o	7	9	9		100

特定のPDBの結果を表示するようにレポートを構成することもできます。

```
set lines 85
col Service_name format a30 trunc heading "Service" break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select con_id, total_requests,
total_calls,total_protected,total_protected*100/NULLIF(total_calls,0) as
Protected
from(
select * from
(select s.con_id, s.name, s.value
    FROM GV$CON_SYSSTAT s, GV$STATNAME n
    WHERE s.inst_id  = n.inst_id
    AND s.statistic# = n.statistic#
    AND s.value != 0 )
pivot(
    sum(value)
    for name in ('cumulative begin requests' as total_requests, 'cumulative end requests' as
Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user calls protected by
Application Continuity' as total_protected)
))
order by con id;
```

次のように表示されます。

CON_ID	Requests	Calls in requests	Calls Protected	Protected %
854	70	283	113	39.9

また、特定の期間を指定できます。次の例では3日間を指定しています。

```
set lines 85
col Service_name format a30 trunc heading"Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select a.instance_number,begin_interval_time, total_requests, total_calls, total_protected,
total_protected*100/NULLIF(total_calls,0) as Protected
from(
select * from
(select a.snap_id, a.instance_number,a.stat_name, a.value
    FROM dba_hist_sysstat a
    WHERE a.value != 0 )
pivot(
    sum(value)
    for stat_name in ('cumulative begin requests' as total_requests, 'cumulative end requests' as
Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user calls protected by
Application Continuity' as total_protected)
)) a,
dba_hist_snapshot b
where a.snap_id=b.snap_id
and a.instance_number=b.instance_number
and begin_interval_time>systimestamp - interval '3' day
```

制限事項

アプリケーション・コンティニュイティを使用する場合は、次の制限事項および考慮事項に注意してください（アプリケーション・コンティニュイティに関する制限および他の考慮事項）。

クライアントの構成

JDBC Thinドライバのチェックリスト

1. 高速接続フェイルオーバー（FCF）と呼ばれるJava向けのFANを構成します。 クライアント・ドライバ12c以降の場合ONSの自動構成で推奨されているURLを使用します。
    - ons.jar、simpleFan.jar（Oracleプールを使用していない場合）（およびオプションのWALLET jar、osdt_cert.jar、osdt_core.jar、oraclepki.jar）がCLASSPATHに存在することを確認します。
    - プール・プロパティまたはドライバ・プロパティfastConnectionFailoverEnabled=trueを設定します
    - サード・パーティのJDBCプールの場合は、Universal Connection Poolが推奨されます
    - ONS用にポート6200を開きます（6200はデフォルト・ポートです。別のポートが選択されている場合もあります）

    推奨の接続文字列を使用できない場合は、以下の設定を使用して手作業でクライアントを構成します。

    oracle.ons.nodes =XXX01:6200, XXX02:6200, XXX03:6200

アプリケーション・コンティニュイティ向けのJDBC Thinドライバのチェックリスト

1. プロパティ・ファイルまたはコンソールでOracle JDBC再実行データソースを構成します

19c DBRUクライアントJARを使用するアプリケーションの場合は、oracle.jdbc.replay.OracleDataSourceImplをスタンドアロンで使用するか、それらをOracle UCPのようなJava接続プール、またはWebLogic AGL Server接続プールのコネクション・ファクトリ・クラスとして構成します。

19c UCPドキュメントでは、Oracle UCP上でAC/TACを有効化する方法について説明しています（上記のJDBCドライバ・データソース・クラス"oracle.jdbc.replay.OracleDataSourceImpl"をOracle UCPデータソースPoolDataSourceImpl上のコネクション・ファクトリ・クラスとして構成します）。

https://docs.oracle.com/en/database/oracle/oracle-database/19/jjucp/application-continuity-using-ucp.html#GUID-EA541BD8-7B19-4A28-BF29-C4A623B41EEC

21c以降のDBRUクライアントJARを使用するアプリケーションの場合は、AC/TACがサービス上にあるときに自動的に有効化するデータソースを使用します。datasource=*oracle.jdbc.datasource.impl.OracleDataSource*（*21.1*以降で使用可能）

WebLogic Serverの場合、Oracle WebLogic Server管理コンソールを使用して、次のローカルの再実行ドライバを選択します。
Active GridLinkアプリケーション・コンティニュイティ接続向けのOracle Driver（Thin）

2. JDBCステートメント・キャッシュを使用します
   アプリケーション・サーバーのステートメント・キャッシュの代わりに、JDBCドライバのステートメント・キャッシュを使用します。これにより、ドライバはステートメントが取り消されることを認識できるようになるため、リクエストの終了時にメモリを解放できます。
   JDBCステートメント・キャッシュを使用するには、接続プロパティoracle.jdbc.implicitStatementCacheSize（OracleConnection.CONNECTION_PROPERTY_IMPLICIT_STATEMENT_CACHE_SIZE）を使用します。キャッシュ・サイズの値はopen_cursorsの数と同じです。たとえば、oracle.jdbc.implicitStatementCacheSize=*nnn*のように指定します。ここで*nnn*は通常、50～200の値であり、アプリケーションで維持されているオープン・カーソルの数と同じになります。
3. ガベージ・コレクタをチューニングします
   多くのアプリケーションでは、ガベージ・コレクタのデフォルト・チューニングで十分です。大量のデータを返すアプリケーションや保持するアプリケーションの場合は、2 G以上といった高い値を使用できます。たとえば、次のとおりです。
   java -Xms3072m -Xmx3072m
   Javaの初期ヒープ・サイズ（ms）のメモリ割当てと最大ヒープ・サイズ（mx）のメモリ割当てを同じ値に設定することを推奨します。この設定により、メモリ・ヒープの増加および縮小で、システム・リソースが使用されなくなります。
4. コミットします
   JDBCアプリケーションでAUTOCOMMITを使用する必要がない場合は、アプリケーションまたは接続のプロパティでAUTOCOMMITを無効にします。Apache Tomcat、IBM WebSphere、IBM Liberty、Red Hat WildFly（JBoss）などのサード・パーティ製アプリケーション・サーバーにOracle UCPまたは再実行ドライバが埋め込まれている場合に、この設定は重要です。
   Oracle UCPのPoolDataSource接続プロパティを使用して、autoCommitをfalseに設定します。
   connectionProperties="{autoCommit=false}"
5. JDBC具象クラス – JAR 12.1および12.2ドライバのみに適用
   JDBCアプリケーションについては、非推奨のoracle.sql具象クラス（BLOB、CLOB、BFILE、OPAQUE、ARRAY、STRUCT、またはORADATA）はOracle Application Continuityでサポートされません（MOS Note 1364193.1『*New JDBC Interfaces*』を参照）。アプリケーションに問題がないことをクライアントで認識するには、ORAchk -acchkを使用します。
   Oracle JDBC Thinドライバのバージョン18c以降、JDBC Replay Driverの制限付き具象クラスのリストは次のとおりに削減されています。oracle.sql.OPAQUE、oracle.sql.STRUCT、oracle.sql.ANYDATA。

OCI（Oracle Call Interface）ドライバのチェックリスト（OCIベースのクライアントには、シック・モードのNode.js、Python、SODAが含まれます）

1. OCIベースのアプリケーションにFANを使用するには、以下を実行します。

- サービスでaq_ha_notificationsを設定します

ONSの自動構成で推奨されている接続文字列を使用します

oraaccess.xmlでauto_config、events、およびwallet_location（オプション）を設定します

```
<default_parameters>
        （その他の設定がこのセクションに含まれる場合があります）
    <ons>
        <auto_config>true</auto_config>
        <wallet_location>/path/onswallet</wallet_location>
    </ons>
    <events>
        True
    </events>
</default_parameters>
```

- オープンソースを含む多くのアプリケーションはスレッド化されます。スレッド化されない場合は、アプリケーションをO/Sクライアント・スレッド・ライブラリにリンクさせます
- ONS用にポート6200を開きます（6200はデフォルト・ポートです。別のポートが選択されている場合もあります）。
  推奨の接続文字列を使用できない場合は、以下のように手作業でクライアントを構成します。
  ネイティブ設定を使用しないOracle Call Interface（OCI）クライアントでは、oraacces.xmlファイルを使用し、eventsをtrueに設定できます。

Python、Node.js、およびPHPではネイティブ・オプションを使用できます。PythonとNode.jsでは、接続プールの作成時にイベント・モードを設定できます。

PHPでは、php.iniを編集してエントリoci8.events=onを追加します。SQL*PlusではFANがデフォルトで有効になります。

アプリケーション・コンティニュイティ向けのOCIドライバについての考慮事項

サポートされるステートメントの全一覧については、ドキュメントを参照してください。OCIStmtPrepareをOCIStmtPrepare2に置き換えます。OCIStmtPrepare()は12.2以降、非推奨になっています。すべてのアプリケーションでOCIStmtPrepare2()を使用する必要があります。TACおよびACでは、OCIStmtPrepare()およびサポートされないその他のOCI APIが許可されますが、これらの文は再実行されません。

https://docs.oracle.com/en/database/oracle/oracle-database/19/lnoci/high-availability-in-oci.html#GUID-D30079AC-4E59-4CC3-86E8-6487A4891BA2

https://docs.oracle.com/en/database/oracle/oracle-database/19/lnoci/deprecated-oci-functions.html#GUID-FD74B639-8B97-4A5A-BC3E-269CE59345CA

ODP.NET管理対象外のプロバイダ・ドライバのチェックリスト

1. すべての推奨パッチがクライアントに適用されていることを確認します。MOS Note『Client Validation Matrix for Draining and Application Continuity』（Doc ID 2511448.1）を参照してください。
2. OCIベースのアプリケーションにFANを使用するには、以下を実行します。

- サービスでaq_ha_notificationsを設定します
- ONSの自動構成で推奨されている接続文字列を使用します
- oraaccess.xmlでonsConfigとwallet_location（オプション）を設定します

ONS用にポート6200を開きます（6200はデフォルト・ポートです。別のポートが選択されている場合もあります）

- 接続文字列でFANを設定します

  "user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true;"

- （オプション）次の接続文字列で、ラインタイム・ロードバランシングを設定します

  "user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true; load balancing=true;"

ACCHKを使用した保護状態の詳細確認

ACCHKは、Oracle Database 19c RU11（19.11）以降のデータベース機能です。データベース・ビューとPL/SQLベースのレポートに、フェイルオーバーにおけるアプリケーションの保護レベルが表示されます。十分に保護されていないアプリケーションはACCHKによって特定され、保護されていない理由が突き止められ、保護状態を改善する方法が提示されます。

また、アプリケーション・コンティニュイティ・データを使用してワークロードのカバレッジを収集し、リクエストに従って詳細情報を提供します。カバレッジを収集するには、データベース・ワークロードを実行する前に、ACCHKを有効にする必要があります。ACCHKでは、フェイルオーバーの診断も実行されます。

詳しくは、『Oracle *RAC*管理およびデプロイメント・ガイド』を参照してください。

ACCHKの有効化

acchkを使用するには、（SQL*Plusなどを使用して）データベースに接続し、以下のコマンドを実行します。

- 次のようにACCHK_READロールを使用して、アプリケーション・コンティニュイティの保護チェック・レポートとビューを実行するユーザーに読取りアクセス権を付与します。

```
GRANT ACCHK_READ to myUser;
```

- 次のようにdbms_app_cont_admin.acchk_setプロシージャを使用して、アプリケーションでアプリケーション・コンティニュイティ・トレースを有効にします。

```
EXECUTE dbms_app_cont_admin.acchk_set(true);
```

デフォルトでは、acchkは600秒後に無効になります。acchk_setプロシージャに値を指定することで、別のタイムアウト時間を設定できます。
たとえば、300秒後に無効にするには、次のように指定します。

```
EXECUTE dbms_app_cont_admin.acchk_set(true, 300);
```

acchkを手作業で無効にするには、以下のプロシージャを実行します。

```
EXECUTE dbms_app_cont_admin.acchk_set(false);
```

手作業で無効にする場合、新しいセッションのみが影響を受けます。現在のセッションでは、セッションが終了するまでトレースは無効にならないことに注意してください。

acchkを有効にした状態で、アプリケーションの機能を実行してみてください。障害を誘発したり、メンテナンス・タスクを実行したりする必要はありません。通常の実行時操作で十分です。アプリケーションの機能を実行したら、acchkを無効にし、組込みのレポートまたはビューのデータを精査します。

ACCHKのレポートとビューの精査

アプリケーションのデータベース操作の実行後、次のようにacchkレポートを精査します。

```
EXECUTE dbms_app_cont_report.acchk_report(dbms_app_cont_report.SUMMARY)
```

レポート・レベルはFULL、WARNING、SUMMARYです。デフォルトのレポートはSUMMARYです。

以下にレポートのサンプルを示します。

```
------------------------------------
------------ ACCHK Report ------------
------------------------------------
CON_ID SERVICE       FAILOVER PROTECTED_ PROTECTED_ REQUESTS AVG_CALLS/ PROTECTED_  AVG_TIME/  PROTECTED_TIME/ EVENT_  ERROR_ PROGRAM   MODULE            ACTION    SQL_ CALL      TOTAL
                              CALLS %   TIME %              REQUEST    CALLS/REQUEST REQUEST MS REQUEST MS       TYPE    CODE                                       ID
------ ------------- -------- -------- -------- -------- --------- ------------ ---------- ---------------- ------- ------ --------- ----------------- --------- ---- --------- -----
3      srv_tacr_pdb1 AUTO     98.734   98.432   117      9.453     9.333        2279.751   2244.014         DISABLE 41409  JDBC Thin AddCustNewOrder   Action-20       COMMIT    1
3      srv_tacr_pdb1 AUTO     98.734   98.432   117      9.453     9.333        2279.751   2244.014         REPLAY_ 41412  JDBC Thin InsertNewChecksum Action-1        SQL/PLSQL 1
                                                                                                            FAILED         Client                                     Execu
End of report.
```

複数のビューが提供され、ビューに問合せを発行してデータを取得できます。DBA_ACCHK_EVENTS、DBA_ACCHK_EVENTS_SUMMARY、DBA_ACCHK_STATISTICS、DBA_ACCHK_STATISTICS_SUMMARYのビューがあります。

付録

継続的な可用性に関する開発者向けベスト・プラクティス

以下は、開発者が可用性の高いアプリケーションを記述するためのベスト・プラクティスです。

接続プールへの接続の返却

開発者のもっとも重要なプラクティスは、各リクエストの終了時に接続を接続プールに返却することです。これは、アプリケーションを最高のパフォーマンスで実行するため、実行時やメンテナンス時のドレイニング作業とリバランシング作業のため、フェイルオーバー・イベントの処理のために重要です。接続の保持がパフォーマンスを改善するという誤った考え方のアプリケーションが一部に存在します。接続を保持しても、パフォーマンス面、スケーラビリティ面のいずれのメリットもありません。あるお客様によると、接続をプールに返却することにより、中間層CPUが40 %削減され、スループットが向上したということです。

リクエスト間のセッション状態の消去

RESET_STATEは、Oracle Databaseのもっとも有益な開発者向け機能の1つです。

アプリケーションによって接続が接続プールに返却されても、フェッチ状態のカーソルとそのセッションで設定されたセッション状態は、消去するための操作を行わない限りそのまま残ります。たとえば、アプリケーションが接続プールとの間で接続の借用と返却を行う場合、セッション状態をクリーンアップしなければ、同じ接続が次に使用されるときに、前のセッション状態が見えることになります。リクエストの終了時には、後でそのデータベース・セッションを再利用する際にリークを避けるために、カーソルをステートメント・キャッシュに戻し、アプリケーションに関連するセッション状態を消去するのがベスト・プラクティスです。

Oracle Database 21c以前のバージョンでは、PL/SQLグローバル変数を消去するにはdbms_session.modify_package_state (dbms_session.reinitialize)を、一時表を消去するにはTRUNCATEを使用します。また、コンテキストを消去し、カーソルをステートメント・キャッシュに戻して取り消すには、SYS_CONTEXT.CLEAR_CONTEXTを使用します。

Oracle Database 21cでは、サービス属性RESET_STATEによって、アプリケーションが設定したセッション状態を消去します。コードは不要です。サービスに対しRESET_STATEをLEVEL1に設定すると、リクエストの明示的な終了時にセッション状態をリセットします。RESET_STATEは、TACによって検出された黙示的なリクエスト境界には適用されません。RESET_STATEを使用する場合、アプリケーションは、リクエストの終了時に状態がリセットされていると信頼して動作できます。

TACを使用する場合、セッション状態を消去すると保護状態が向上します。TACはより高頻度で再有効化できます。Oracle Database 23cでは、**RESET_STATE**は、TACを使用するすべてのアプリケーションとTACを使用しないすべてのアプリケーションで使用できます。

PL/SQLにおけるCOMMITの埋め込み回避、およびCOMMIT ON SUCCESSとAUTOCOMMITの回避

トップレベルのコミット（OCOMMIT、COMMIT()、またはOCITransCommit）を使用することを推奨します。アプリケーションが、PL/SQLに埋め込まれたCOMMITを使用している、またはAUTOCOMMIT、COMMIT ON SUCCESSを使用している場合、停止またはタイムアウトの後のリカバリができない可能性があります。PL/SQLはリエントラントではありません。PL/SQL内のコミットが実行されると、そのPL/SQLブロックは再送信できません。アプリケーションは、データを読み取り済みの可能性があるとして、正常でないコミットを選択しないようにする必要があります。または、バッチ処理の場合は、チェックポイントと再起動の手法を使用する必要があります。AUTOCOMMITまたはCOMMIT ON SUCCESSを使用する場合、返されたメッセージは失われます。

アプリケーションがトップレベルのコミットを使用している場合は、透過的アプリケーション・コンティニュイティ（TAC）、アプリケーション・コンティニュイティ（AC）、およびTAF Select Plus（12.2）が完全にサポートされます。アプリケーションで、PLSQLに埋め込まれたCOMMITを使用している、またはAUTOCOMMITもしくはCOMMIT ON SUCCESSを使用している場合、COMMITを含む呼出しの実行が完了しなかったときは再実行できないことがあります。

問合せでのORDER BYまたはGROUP BYの使用

アプリケーション・コンティニュイティによって、再実行時にアプリケーションには確実に同じデータが表示されます。同じデータをリストアできない場合、アプリケーション・コンティニュイティでは再実行が許可されません。SELECTでORDER BYまたはGROUP BYが使用されている場合は、順序が保存されます。Oracle RAC環境では、問合せオプティマイザはほとんどの場合同じアクセス・パスを使用するため、同じ順序で結果が表示されます。アプリケーション・コンティニュイティはまた、AS OF句が許可されている場合はどこでも、内部でAS OF句を使用して同じ問合せ結果を返します。

SQL*Plusの考慮事項

SQL*Plusは通常、何かを試す際の主要ツールです。当然ながらSQL*Plusは、本番環境で使用される実際のアプリケーションを反映していないため、常に本物のアプリケーション・テスト・スイートを使用してフェイルオーバー計画をテストし、保護レベルを測定した方が良いでしょう。SQL*Plusはプールされるアプリケーションではないため、明示的なリクエスト境界はありません。SQL*Plusをレポートなどで使用するアプリケーションもあります。フェイルオーバーと一緒にSQL*Plusを使用する場合は、以下を確認してください。

1. FANはSQL*Plusでは常に有効化されています。ONSエンド・ポイントを自動構成する推奨の接続文字列（上記参照）を使用します。
2. SQL*Plusを使用する上で重要なのは、データベースに対するラウンドトリップを最小限に抑えることです（https://blogs.oracle.com/opal/sqlplus-12201-adds-new-performance-features）。
3. SQL*PlusはOracle 19c以降、TACでサポートされています。最適な結果を得るには、arraysizeを大きい値に設定します（例：set arraysize 1000）。serveroutputを有効化しないでください。有効化すると、リストアできないセッション状態が生成されますOracle Database 23cには、この制限はありません。
4. SQL*PlusはOracle 12.2以降、ACでサポートされています。ACには黙示的な境界がないため、最初のコミット後に再有効化されません。

エンド・ツー・エンド（e2e）のトレース

Oracle Enterprise Managerの上位コンシューマ、TKPROF、アプリケーション・コンティニュイティの統計情報、ACCHKなどは、サービスごとに分離されるほか、モジュールやアクションごとにも分離されます。各種サービスを使用するでしょうが、アプリケーションではモジュールとアクションを使用して作業を指定するのが良いプラクティスです。モジュールとアクションのタグを使用している場合、Oracle Enterprise Manager（Oracle EM）の上位コンシューマ、AWR、統計情報、ACCHKでレポートされます。モジュールとアクションを設定するには、PL/SQLパッケージのDBMS_APPLICATION_INFOではなく、ドライバが提供するAPIを使用します。APIはローカル・ドライバ・コールであるため、パフォーマンスがより優れているためです。

ORACLE

アプリケーションとサーバーのタイムアウトの整合

アプリケーション・レベルのタイムアウトが、基盤システムの検出およびリカバリのための時間として設定されているタイムアウトより短いと、基盤となるリカバリを完了するために使用できる時間が不足します。タイマーの整合性が保たれていないと、システムのリカバリが完了する前にアプリケーション・コンティニュイティによる再実行が開始され、成功するまで再実行が繰り返し試行される可能性や、リクエストがタイムアウトしてアプリケーションまたはユーザーにエラーが返される可能性があります。

リソース・マネージャは、長時間実行されているSQLを停止、隔離、または降格するため、およびSQLが最初から実行されないようにするための、オラクルによる推奨機能です。ハングしたシステムや完全に停止したシステムへの対応のためにREAD_TIMEOUTを使用する場合は、READ_TIMEOUTをリカバリ・タイムアウトより大きい値にする必要があります。小さい値のREAD_TIMEOUTを使用することは推奨されません。これにより、リカバリ時に障害が発生しなかったシステムに不要な中断が生じる可能性があります。

アプリケーションでREAD_TIMEOUTやHTTP_REQUEST_TIMEOUT、または何らかのカスタム・タイムアウトを使用することを検討してください。使用する場合は、次の指針を適用してください。

READ_TIMEOUT > EXADATAの特別なノード・エビクション（FDDN）（2秒）

READ_TIMEOUT > MISSCOUNT（デフォルトは30秒、12c Grid Infrastructureでは変更可能）

READ_TIMEOUT > Data Guard Observer：FastStartFailoverThreshold（デフォルトは30秒、変更可能）

FastStartFailoverThreshold > MISSCOUNT（2回以上）

READ_TIMEOUT > FAST_START_MTTR_TARGET（多くのシステムでは15秒または30秒を選択）
READ_TIMEOUT > Oracle SQL*NET level: (RETRY_COUNT+1) * RETRY_DELAY

READ_TIMEOUT < Replay_Initiation_Timeout（サービス上で変更可能、デフォルトで300秒）

リクエストのキャンセルが早すぎないようにするため、アプリケーションのタイムアウト値を次の最大値より大きい値にする必要があります。
(MISSCOUNT（またはFDNN） + FAST_START_MTTR_TARGET)、または
(FastStartFailoverThreshold + FAST_START_MTTR_TARGET + 開く時間)

ORACLE

元の値を維持するための付与の追跡

データベースでKEEPに対してどの付与が設定されているかを把握するには、次のようなSQLを使用します。

ALTER SESSION SET CONTAINER=&PDB_NAME ;

```
set pagesize 60
set linesize 90

ttitle "Sequences Kept for Replay"
col sequence_owner format A20 trunc heading "Owner"
col sequence_name format A30 trunc heading "Sequence Name"
col keep_value format A10 trunc heading "KEEP"
break on sequence_owner

select sequence_owner, sequence_name, keep_value
from all_sequences, all_users
where sequence_owner =
username and oracle_maintained
= 'N'
order by sequence_owner, sequence_name;

ttitle "Date/Time Kept for Replay"
col grantee format A20 trunc heading "Grantee"
col PRIVILEGE format A20 trunc heading "Keep"
col ADMIN_OPTION format A8 trunc heading "Admin|Option"
break on grantee

select grantee, PRIVILEGE, ADMIN_OPTION
from dba_sys_privs, all_users
where
        grantee = username
and oracle_maintained = 'N'
and PRIVILEGE like '%KEEP%'
union
select distinct grantee, 'NO KEEP' PRIVILEGE, 'NO' ADMIN_OPTION
from dba_sys_privs l1, all_users
where
        grantee = username
and oracle_maintained = 'N'
and l1.grantee not in
         ( select l2.grantee
         from dba_sys_privs l2
         where PRIVILEGE like '%KEEP%' )
order by privilege, grantee;
```

次の例のような形式でデータが返されます。

```
Tue Nov 16                                                    page       1
                          Sequences Kept for Replay

Owner                    Sequence Name                          KEEP
------------------------ -------------------------------------- ------------
MOVIESTREAM              MDRS_1715A$                            Y
                         MDRS_17165$                            Y


Tue Nov 16                                                    page       1
                          Date/Time Kept for Replay

                                                                Admin
Grantee                  Keep                                   Option
------------------------ ---------------------------------------- ----------
ADMIN                    NO KEEP                                NO
GGADMIN                  NO KEEP                                NO
MOVIESTREAM              KEEP                                   NO
RMAN$VPC                 NO KEEP                                NO
```

高速アプリケーション通知のデバッグ

FANイベントの送信には次の2つのプロトコルが使用されます。

- Oracle Notification Server（ONS）または完全FAN
- 帯域内FAN

ONS経由でFANイベントを受信するには、イベントをパブリッシュしているONSデーモンに対してサブスクライブする必要があります。もっとも単純な構成では、ONSデーモンはGrid Infrastructure内にインストールされ、データベース層を形成しているクラスタ上で実行されます。

ONSのデフォルト・ポートは6200です。次のように実行して確認できます。

```
$ srvctl config nodeapps
```

ONSの構成が出力されます。

```
ONS exists:Local port 6100, remote port 6200, EM port 2016, Uses SSL true
ONS is enabled
ONS is individually enabled on
nodes:ONS is individually disabled on
nodes:
```

このリモート・ポートが、イベントが送信されるポートです。このポートは、構成対象となるファイアウォールを経由するTCP POST用に開いておく必要があります。

ONSへのサブスクリプションがクライアント・アプリケーションによってオープンされたことを確認する場合、GIクラスタへのアクセス権があるときは、実行時にONSデーモンへの接続を精査します。

```
$ onsctl debug
```

以下の情報が表示されます。

リモート・ポートとローカル・ホストのIPアドレス（この例では6200）

実行時に接続しているクライアント（接続中のクライアントのIPが表示される）

```
                  IP ADDRESS          POR       TIME SEQUENCE FLAGS
---------------------------------- ----- -------- -------- --------
                  123.123.1.123 6200 6205af70 00000008 00000008

<<< Information removed >>>

Client connections:(5)
        ID                CONNECTION ADDRESS                PORT      FLAGS     SNDQ   REF    PHA    SUB
---------- ---------------------------------------- ---------- ---------- ------          ----
         0                                internal          0   000044a         0          1 IO      1
         3                               127.0.0.1      16576   040041a         0          1 IO      0
         d                               127.0.0.1      16598   040041a         0          1 IO      1
         e                               127.0.0.1      16600   040041a         0          1 IO      1
         f                   ::ffff:100.101.102.103      61838   04c042a         0          1 IO      1
```

クライアント側のFAN用の設定は適切ですか。

- Universal Connection PoolとJDBC Thinドライバの場合は、それらの使用時にプール型のプール・プロパティFastConnectionFailoverEnabled = trueが設定されていることを確認します。
- ODP.Netの場合は、接続文字列にpooling=true; HA events=true が設定されていることを確認します。
- OCIクライアントの場合
    - oraaccess.xmlに“<events>true</events>”を設定します。
    - 動的データベース・サービスで次のようにして通知を有効にします。srvctl modify service –notification TRUE
- WebLogic Active Grid Linkの場合、FANはデフォルトで有効になっています。これは管理コンソールで確認できます。ONSエンド・ポイント向けに、管理コンソールでons.configurationも設定してください。

その他の資料

アプリケーション・コンティニュイティに関するOracle Technology Network（OTN）ホームページ
http://www.oracle.com/goto/ac
アプリケーション・コンティニュイティ

『アプリケーション・コンティニュイティの確保』（*https://docs.oracle.com/en/database/oracle/oracle-database/21/racad/ensuring-application-continuity.html#GUID-C1EF6BDA-5F90-448F-A1E2-DC15AD5CFE75*）

『*Graceful Application Switchover in RAC with No Application Interruption*』
My Oracle Support (MOS) Note：Doc ID 1593712.1

JAVAアプリケーション・サーバーにUCPを埋め込む方法

「*WLS UCP Datasource*」https://blogs.oracle.com/weblogicserver/wls-ucp-datasource

『*Design and Deploy WebSphere Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP*』（http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-websphere-2409214.pdf）

「*Reactive programming in microservices with MicroProfile on Open Liberty 19.0.0.4*」
（https://openliberty.io/blog/2019/04/26/reactive-microservices-microprofile-19004.html#oracle）

『*Design and deploy Tomcat Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP*』（http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-tomcat-2265175.pdf）

高速アプリケーション通知

http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/fastapplicationnotification12c-2538999.pdf

ORACLE

Connect with us

+1.800.ORACLE1までご連絡いただくか、**oracle.com**をご覧ください。北米以外の地域では、**oracle.com/contact**で最寄りの営業所をご確認いただけます。

blogs.oracle.com facebook.com/oracle twitter.com/oracle